# 並列計算機ADINAと並列アルゴリズム (数値計算のアルゴリズムの研究)

AUTHOR(S):

野木, 達夫

CITATION:

野木, 達夫. 並列計算機ADINAと並列アルゴリズム (数値計算のアルゴリズムの研究). 数理解析研究所講究録 1982, 453: 42-59

ISSUE DATE:

1982-02

URL:

http://hdl.handle.net/2433/102993

数理解析研究所講究録
453巻 1982年 42-59



# 並列計算機ADINAと並列アルゴリズム

京大 工学部 野木達夫

## 1. はじめに

核融合，高速流体，気象など複雑な自然現象のシミュレーションには，超高性能計算機が欠かせなくなってきている。既に，これらの計算のために開発されたスーパーコンピューターとして，ILLIAC-Ⅳ，STAR-100，CRAY-I 等が稼動している。

これらの計算機では，超高速化のために，何んらかの並列演算方式を採用しているが，大別して，マルチファンクション方式，パイプライン方式，パラレルプロセッサ方式がある。前の2方式については，すでに，大型汎用計算機でも，広く利用されているところであるが，並列度に限界があり，大中なスピードアップにはつながらない。

一方，パラレルプロセッサ方式では，並列度に原理的な制約はない。並列処理にあたって，前の2方式の場合のように

相互にきちっと結ばれた形でなく，多数のプロセッサで，同時に処理する形をとるので，柔軟性に富む。しかも，速くて安価なLSIプロセッサの出現が，パラレルプロセッサ方式による大規模な複合計算機の可能性を伺わせている。

現時点で，最も超高速が要求される科学技術計算に対処しうるマシンを考えるならば，主として，シミュレーション用という専用性の中で，望しいプロセッサアレイのアーキテクチャを求め，各プロセッサレベルでは，可能な限り，マルチファンクション，パイプライン方式を取り入れて高性能化をはかるのが至当といえよう。このことを基本におさえ，従来のものをはるかに超える性能を目標にして考案したのが，ADINA Computer である。

## 2. シミュレーションにおける並列計算

自然現象のシミュレーションの場合，通常，空間において同時的に進行する現象は，コンピューターのデータメモリの中味（物理変数）の時間的変化でもって表現される。普通のユニプロセッサコンピューターでは，この変化を規定するものは，一本のシリアルなプログラムであり，基本的には，時間を止めては，空間領域（データメモリ）上をシリアルに辿るアルゴリズムの反復過程である。この完全なシリアル処理

が，計算速度を低くおさえるものであり，少しでも同時的処理部分を増す努力が並列計算に他ならない。

さて，ユニプロセッサの場合，データメモリの読み書きが，‘窓口’一つでなされることで全く支障はない。しかし，1セットのデータメモリを共有するマルチプロセッサシステムで，あるパラレルプログラムを実行する場合には，一つの‘窓口’が正に隘路となって，メモリアクセスの競合が生じ，並列計算がスムースに進行しなくなる。

この競合を緩和するために，同時にアクセスできる複数のデータメモリセットを設け，適当なインターコネクションネットワークで，アクセスの交通整理を行う方法が採られる。しかし，この場合のコントロールは，一般に複雑で，しかも決して競合がなくなるものではない。共有メモリを利用する場合の泣きどころである。

そこで，共有データメモリの利用を避けるものとすれば，空間領域の写しは，各プロセッサの私メモリに分有されることになる。この場合には，新たに，プロセッサ間のデータ交換のためのバス接続の問題が浮上してくる。

いま，仮に，領域の代表点（格子点）の個数だけのプロセッサを用意し，代表点毎に，対応したプロセッサによって計算を行い，すべての点で，同時的に決定していく方法をと

たとする。この場合には，一見，自然現象に類似した進行過程が計算機上で再現されることになる。ただし，このような計算を許すアルゴリズムは，非定常問題に対するエクスプリシット法や，定常問題に対する単純な反復法に限られる。これらの方法を実現するためには，通常，隣接プロセッサ間でデータ交換が可能になる方途さえもっておればよい。この方式は，ハードウェア量が比較的に少いので，これまでのプロセッサアレイ方式の標準型になっている。ILLIAC Ⅳや，PACS（筑波大．星野氏ら）は，まさに，この考えを実現したもので，2次元問題に，2次元プロセッサアレイを対応させることを基本にしている。

なお，ここで留意すべき点をあげると，第1に，空間領域の代表点数に相当するプロセッサを備えることは，実際上不可能であること。そのために，実際には，かなりの数の代表点を一まとめにしたブロック毎に，一台づつのプロセッサを割りあてることになる。そして，各プロセッサが，私メモリ上のデータブロックをシリアルに辿る計算が，すべてのブロックにわたって並列的に行われる形になる。このとき，計算途中で，隣接プロセッサ間でかなりのデータ交換がなされることになる。

第2に，直接的データ交換が，隣接プロセッサ間でしか許

されないので，離れたプロセッサへのデータ転送に手間がかかり，そのために，アルゴリズムとして，エクスプリシット法や，単純反復法だけしか，効率よく利用できないことである。

当然のことながら，アルゴリズムとして優れたものが，効率よく計算されてこそ，最高速性に近づきうる者である。この点からして，インプリシット法なり，直接法の利点を尊重しない法はないので，これらのアルゴリズムが効率よく処理されるマシンであることが望しい。

## 3. エコノミカルスキームと ADINA Computer

差分スキームの研究から生れた重要概念の一つが，エコノミカルスキームである。それは，簡単に言えば，無条件安定なインプリシットスキームであって，各ステップにおける計算量が，エクスプリシットスキームの計算量と同じオーダー（格子点数に関して）であるものであり，全計算時間を短かくするためには，大変好都合である。

そのような一群のスキームを生み出す契機となったのが，ADI法である。これらのエコノミカルスキームの特徴は，1ステップ差分作用素を，幾つかの分ステップ作用素の積で近似することにあり，その際，各分ステップ作用素の逆を求

める計算が比較的容易なものになる様にする。通常，分ステップ作用素は，対応した一次元方向にのみインプリシット性をもつ多数の独立した小問題に分解された形になっている。

立方領域の問題に対しては，3つの分ステップを考え，第1の分ステップでは，各代表対$(y_j, z_k)$毎に独立に，x方向にインプリシット性をもつ作用素の逆を求め，引き続き，第2，第3の分ステップでも同様に，それぞれy方向，z方向にインプリシット性をもつ作用素の逆を求めることになる。つまり，1ステップ進めるために，1次元方向にのみインプリシットなスキームからでてくる小規模の連立方程式を多数組解くことになる。このとき，1方向の処理毎に，独立な連立方程式の組が並列に解き下されうる。ここにADINA Computerの着眼点がある。

ここで，一つ注意すべきことは，上に述べた小規模の連立方程式の係数行列は，多くの場合，三重対角形であって，直接法によって容易に解けるものであること。その際，この三重対角方程式をも並列処理することが考えられるが，その効率は低いものである。([1]) 従って，それらの小規模の連立方程式の各々を解くには，1台のプロセッサを割りあてることにしておくのが望しい。

そうすると，上述の立方領域問題において，各代表対$(y_j,$

$z_k$)毎に1台づつのプロセッサを割りあて，x方向に並んだ格子点の写しを，対応するプロセッサの私メモリーにとるものとする。各プロセッサはその私メモリ上をシリアルに辿る（ここにも先にあげた第1の留意点に相当することがみられる）。その過程が，すべてのプロセッサにわたって並列的に進行することになる。同じプロセッサアレイは，y方向，z方向にも利用できなければならない。それは，別のアレイを用いたのでは効率をおとしてしまうからである。そこで問題になるのが，第2，第3の分ステップへも計算が引き継がれる様に，プロセッサ間のデータ転送の方途を用意することである。この点で，ユニークなアイデアを実現したのが，ADINA Computer である。

以下では，その概要を，アーキテクチャをうまく反映したデータ配列構造でもって説明する。

## 4. ADINA I

簡単のために，ここでは，2次元問題を例にして述べる。それを，象徴的に，2次元配列データ表現

"$u[i,j]$, $i=1,2,\cdots,N$, $j=1,2,\cdots,N$"

で表わすものとする。これに対し，ADINA Computer 上では別表現を与えよう。

2段の分ステップ法を用いるものとし，第1分ステップでは，$j$ 番目のプロセッサが，$u[i,j]$, $i=1,2,\cdots,N$ の値を（$i$ についてシリアルに）更新する役目を担い，その過程がすべての $j=1,2,\cdots,N$ にわたって並列に進行するものとする。このとき，$u[i,j]$, $i=1,2,\cdots,N$ の値は，$j$ 番目のプロセッサの私メモリに保有されるものであるから，そのことを明示するために，記号

$$u[i,(j)],\quad i=1,2,\cdots,N$$

を用いる。すなわち，これは，$j$ 番目のプロセッサがもつ $i$ 方向の一次元配列を意味する。

第2分ステップでは，上に得た値を更新するため，$i$ 番目のプロセッサが，

$$u[i,(j)],\quad j=1,2,\cdots,N$$

の計算を担うことが期待される。しかし，第1分ステップが終った段階では，$i$ 番目のプロセッサが保有するデータは，そのうちの $u[i,(i)]$ のみであり，他の番号 $j$ のものは，他のプロセッサから転送して貰わなければならない。転送されたとして，そのデータを

$$u[(i),j],\quad j=1,2,\cdots,N$$

と表わすことにする。これは，$i$ 番目のプロセッサの私メモリに保有された $j$ 方向の一次元配列を意味する。このデータ

を更新することが，第2分ステップでのi番目のプロセッサの役目である。

以上のことから，ADINA Computer上では，考えている2次元問題を，象徴的に

“ u[i,(j)], u[(i),j], i,j=1,2,…,N ”

と表わすことが自然であることがわかる。ここで，注目すべきことは，通常では一通りのデータ表現 u[i,j] の代りに，二様の表現 u[i,(j)] と u[(i),j] を与えていることである。計算には，( ) で示されたプロセッサが並列的に関与し，各プロセッサは，( ) のついていないインデックスについてシリアルに計算を実行する。上に触れた様に，第1分ステップで取り扱うデータが {u[i,(j)]} で，第2分ステップで扱うのが {u[(i),j]} になるが，このとき，第1分ステップから第2分ステップへの移行時，および，次のステップへの移行時に必要なデータ転送は，それぞれ，代入式

$$u[(i),j] := u[i,(j)],$$
$$u[i,(j)] := u[(i),j]$$

の形で表現される。

これらの転送動作をスムースに進めるために，バッファメモリの2次元配列を備え，(i,j) バッファには，i番目，j番目のプロセッサが接続される様にする。このとき，一方向

の転送は，一方のプロセッサが書きこみ，他方のプロセッサが読みとることで済む。

実は，このような接続方法は，任意の2台のプロセッサ間で，バッファメモリを介するだけでデータ転送が可能な完全結合方式になっていて，プロセッサ台数を増せば，その数の2乗に比例して，ハードウェア量が増え，莫大なものになる。その中でも，比較的ハード量を少くする工夫として，バッファメモリに，シリアルメモリを用いることができる。それは，RAMを用いたときのアドレスライン分だけ省けるからである。このあたりの工夫まで取り入れた機械をADINA Iと呼び，現在，試作中である（[2]，[4]）。

## 5. ADINA II

実際の要求に応えるためには，プロセッサ台数を増し，さらに3次元問題にも対処できなければならないので，ADINA Iのアーキテクチャのままでは具合いが悪い。新たな回答がADINA IIである（[3]，[4]）。

象徴的に表わされた，通常の3次元問題

“ $u[i,j,k]$, $i,j,k = 1, 2, \cdots, N$ ”

に対して，ADINA II上で考える問題は，

“ $u[i,(j,k)]$, $u[(i,j),k]$, $u[i,j,k]$

$i, j, k = 1, 2, \cdots, N$ ”

と表現されるものである。ここに示されたデータ表現は，それぞれ，プロセッサ $(j, k)$，プロセッサ $(k, i)$，プロセッサ $(i, j)$ の私メモリ上の一次元配列を表わしている。すなわち，プロセッサの2次元配列を備えている。

考えるべきことは，データ転送の方途であるが，ADINA II では，代入

$$u[i), j, (k] := u[i, (j, k)],$$
$$u[(i, j), k] := u[i), j, (k],$$
$$u[i, (j, k)] := u[(i, j), k]$$

および，それぞれの逆に限って，許容される様に，バッファメモリアレイを配置する。上の第1式は，$i$ 方向の一次元配列の組を，$j$ 方向の一次元配列の組に編集するときに使われる。これを実現するために，同一の番号 $k$（両辺で，内側の（ ）の中にあるもの）をインデックスに含む プロセッサ $(j, k)$, $j = 1, 2, \cdots, N$ の組と，$(k, i)$, $i = 1, 2, \cdots, N$ の組が，相互に相手側の任意のプロセッサとつながる様な $N \times N$ バッファアレイ（ADINA I 風の）を $k$ 番目のアレイとして，全体で $N$ アレイ備える。上記第2，第3式の転送にも別のバッファセットが必要のように思えるが，実は不必要である。それは，上記のアレイセットさえあれば，プロセ

ッサ $(k,i)$, $k=1,2,\cdots,N$ と プロセッサ $(i,j)$, $j=1,2,\cdots,N$ は, $i$ 番目のアレイにつながり, プロセッサ $(i,j)$ $i=1,2,\cdots,N$ とプロセッサ $(j,k)$, $k=1,2,\cdots,N$ は $j$ 番目のアレイにつながっているからである。

かくして, $N^2$台のプロセッサに対して, $N^3$ケのバッファメモリを設けることで, 上記の転送がすべて可能になる。さらに, ここで注目すべきことは, 一般に, 任意の2台のプロセッサ $(i,j)$ と $(l,m)$ が直接にデータ交換できるわけではないが, 中間にプロセッサ $(j,l)$ を介せば, いつも可能になることである。しかも, どのプロセッサ間でも, それらの番号間の距離に依存しない一様な時間で転送できることである。この事実は, ADINA Ⅱ の汎用利用の道も大きく拓く。

## 6. R-S方式

いままでに触れなかった重大事として, 転送オーバーヘッドの問題がある。ADINA Computer では, その最も特徴とするバッファメモリアレイのハードウェア量を極力少くするために, シリアルメモリの利用を考えるだけに, 一層, 転送がボトルネックに思えてくる。しかし, 実際には, 深刻な問題ではないことを, 以下に説明する。

普遍的な計算のパターンとして, $N^2$台のプロセッサが並列

的に，それぞれ対応したNケのバッファメモリからデータを読みとり（Receive 又は R），私メモリ上で O(N) 量のシリアル計算を行い（Compute 又は C），ついで，Nケのバッファメモリに書き込む（Send 又は S）という基本プロセスを考える。この単位プロセスを簡単に

$$\{ R - C - S \}$$

と表わす。この単位は，並列処理の同期単位になる。従って，Receive と Send の並行動作は考えない。しかし，Receive と Compute および Compute と Send はできるかぎり並行に動作させる。そのためにチャネルを設ける。また，Compute プロセスは，一般に

$$\{ R - M - S \}$$

のパターンを取る。例えば，ガウス消去法（ダブルスイープ法）では，R と S は Reduction と Substitution を意味し，M に相当するものはない。一般的には，R を前処理段階，M を途中計算，S を結果算出段階と解する。このときの R も S も通常，一次元方向未知数 N に対応して O(N) の計算量である。従って，上に現れた2つの R，および2つの S とも同じオーダーの計算量になるので，転送の R と S は，それぞれ Compute の R と S との並行処理（パイプライン的処理）により，後者の R と S に埋め込むことができる。かくして，転送

チャネルスピードをうまく設計することにより，転送オーバーヘッドは殆んど現れないことになる。以上のような転送・計算方式を R-S 方式と名づける。

この R-S 方式は，ADINA Computer が通常は 処理単位として，一次元データ配列上のシリアル計算を行うことを前提にしている。ところで，仮に，完全に並列な計算があったとすると，それは，転送手続きを生のオーバーヘッドとして蒙ることになるであろう。ADINA Computer は シリアル計算を内包するという'禍'を転じて，転送オーバーヘッドをかくすという'福'としているといえる。

この事情から，ADINA II が3次元問題を分ステップ法で解くとき，殆んど 100% の効率に達することが予測できる。すなわち，$N^2$台のプロセッサで，確実に $N^2$ 倍の実効スピードが出せるのであり，これは，従来の並列計算機では到底達し得なかった性能である。

## 7. 補足事項

1) これまで，$N^2$台のプロセッサに対して，丁度，格子点数 $N^3$ の問題を処理する説明をしてきたが，普通は，格子点数の方がもっと多くて $(2N)^3$ や $(3N)^3$ になるであろう。これらの場合にも，使用する私メモリの'深さ'が $2^3$倍，$3^3$

倍になるだけで，並列計算の効率そのものは下らない。

2) ADINA Ⅱが3次元問題に適したアーキテクチャになっていることは明白だが，同時に，2次元問題に対しても，ADINA I 風に利用できることを注意しておく。それには，処理すべき2次元データに，次のように，4次元データを対応させればよい：

$u[p,(q)] \leftrightarrow v[p \bmod N, (q \bmod N, q \operatorname{div} N)][p \operatorname{div} N]$

$u[(p),q] \leftrightarrow v[(p \bmod N, q \bmod N), p \operatorname{div} N][q \operatorname{div} N]$

このとき，転送 $u[(p),q] := u[p,(q)]$ のために，仲介プロセッサを利用することは勿論である。

3) ADINA Computer は，インプリシット法を処理することを念頭において，考案したものであるが，エクスプリシット法に対しても，全く同じく高い効率の処理能力をもっている。この場合にも，3次元問題であれば，3様のデータ $u[\cdot,(j,k)]$, $u[i),\cdot,(k]$, $u[(i,j),\cdot]$ が必要になる。というのは，例えば，1番目のデータだけなら，i方向の差分商の計算ができても，j,k 方向の差分商を手にするに足るものでないからである。

4) 通常の行列演算，連立方程式の解法（CG法など），FFT計算，さらには シミュレーション用の粒子コードにも，高効率で利用できる（[2], [4]）。

5) 二重括弧つきのデータ配列表現は，計算の分担，並列演算の指示を簡明に与えるものであり，従来の高水準言語プログラムからそう離れずにプログラムできるものである。

6) データ交換のためのハードウェア量で比較すれば，ADINA II の $O(N^3)$ は，$N^2$台のプロセッサをもつ ILLIAC IV の $O(N^2)$ と，ADINA I やクロスバ方式の $O(N^4)$ の中間に位置する。

## 8. まとめ

ADINA Computer は，2次元，3次元シミュレーションに大変有用である。そのことは，幾つかのアイデアに基いている。

i) 各プロセッサは1次元問題を解く役目を担う。

ii) プロセッサ間のデータ転送のためにバッファメモリアレイを設ける．

iii) プロセッサ群とバッファアレイの複合体は，各方向のサブプロブレムを解くために，シチュエーションを変える。

iv) 内部括弧を用いた新しい型のデータ配列表現を導入し，並列処理プログラムを書く。

ADINA Computer を一言でいえば，'内部括弧( )を含

んだマシン'とでもいえよう。

この小文では、アーキテクチャについても、応用側についても、殆んど触れられなかったが、全体についてまとめたものとして[5],[6]の文献が参考になる。

## 参考文献

[1] Stone, H.S., Parallel Tridiagonal Equation Solvers, ACM Transactions on Mathematical Software, Vol.1, No.4 (1974) 289-307.

[2] Nogi, T. and Kubo, M., ADINA Computer I, I. Architecture and Theoretical estimates, Memoirs of the Faculty of Engineering, Kyoto University, 42(4) (1980) 421-439.

[3] Nogi, T., ADINA Computer II, I. Architecture and Theoretical estimates, ibid., 43(1) (1981) 124-144.

[4] Nogi, T., ADINA Computer I and II, II. Data Structure, ibid., 43(3) (1981) 434-450.

[5] 野木達夫, ADINA Computer -GFLOPSマシンへの挑戦-, 第75回情報工学研究談話会, 京大情報工学教室 (1981)

[6] Nogi, T., Parallel Machine ADINA, Fifth International Conference on Computing Methods in Applied Sciences and Engineering : Edited by R. Glowinski and J. L. Lions, North-Holland Publishing Company (to appear).